



**Actividad 10 Comparacion de Algoritmos de Agrupacion**

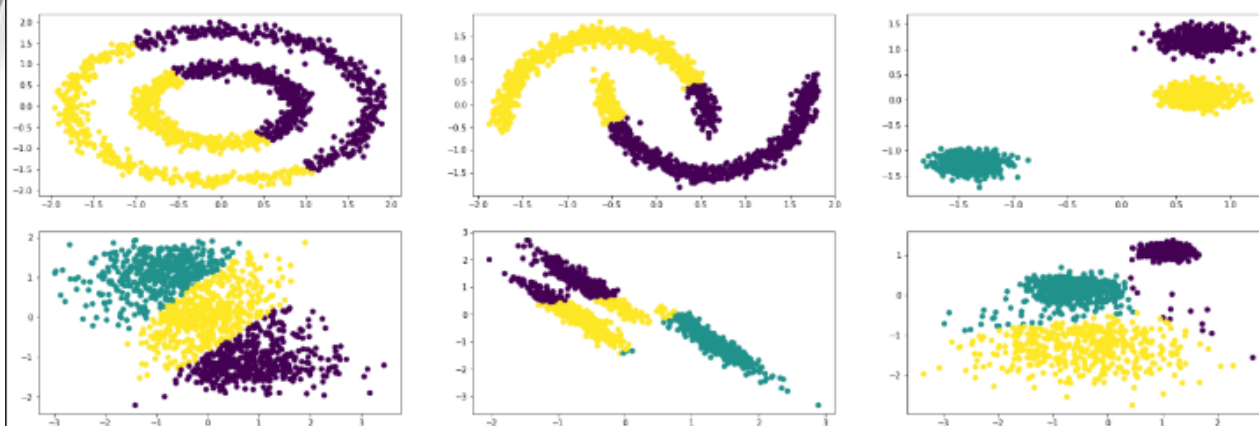
**Alumno: José Osvaldo Farfán de León**

**Materia: Seminario de IA II**

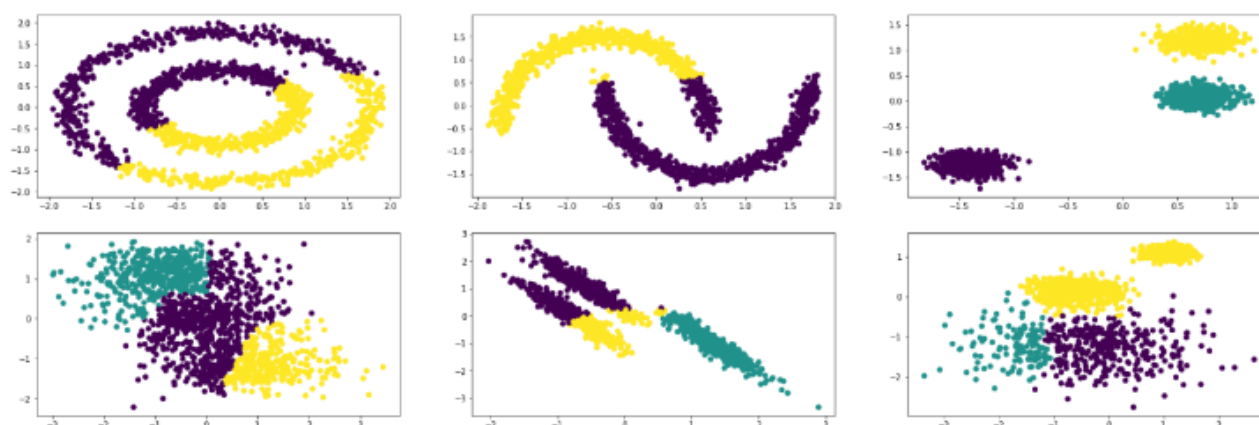
**Profesor: Carlos Alberto Villaseñor Padilla**

**Sección: "D04"**

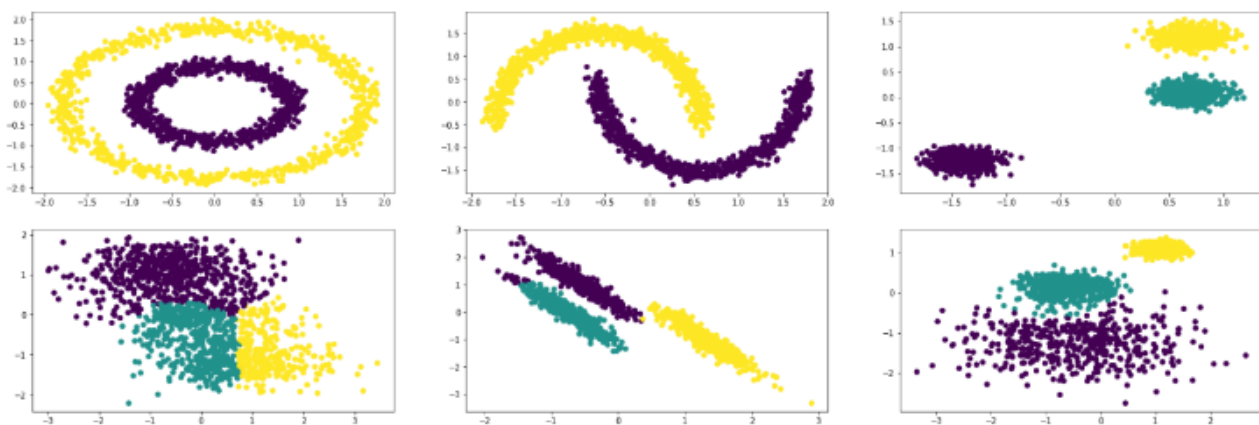
## KMeans



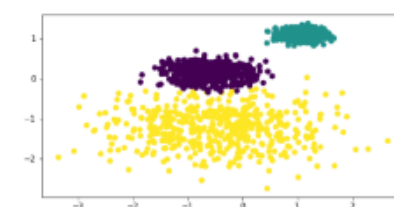
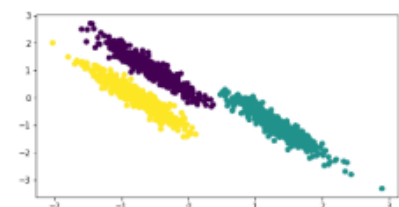
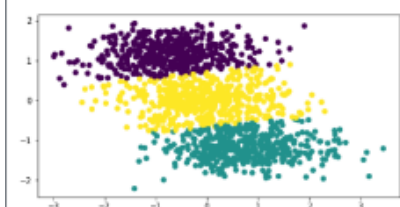
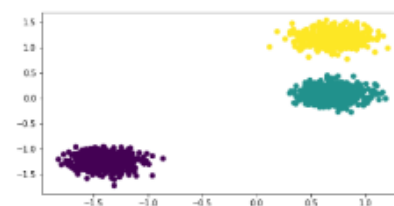
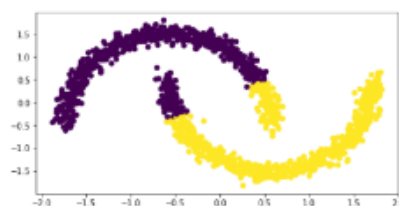
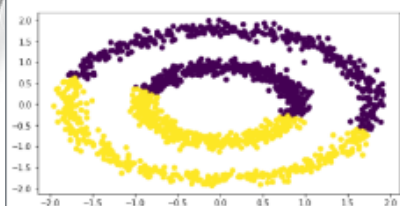
## Birch



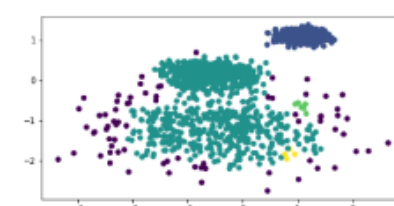
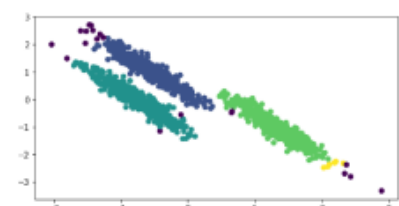
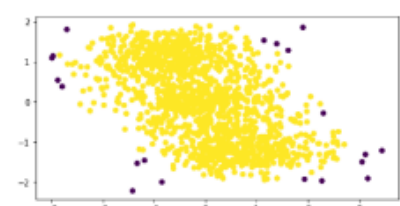
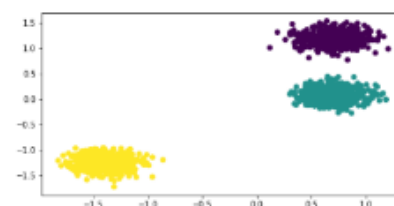
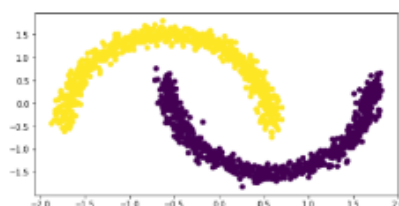
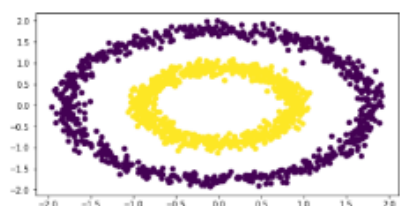
## SpectralClustering



## GaussianMixture



## DBSCAN



```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn import cluster, datasets, mixture, metrics
5 from sklearn.preprocessing import StandardScaler
6
7 #creacion de datasets-----
8 np.random.seed(0)
9 n_samples = 1500
10 X = 6 * [None]
11
12 #circulos concentricos
13 xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
14 X[0] = StandardScaler().fit_transform(xtemp)
15
16 #lunas
17 xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
18 X[1] = StandardScaler().fit_transform(xtemp)
19
20 #Blobs
21 xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
22 X[2] = StandardScaler().fit_transform(xtemp)
23
24 #PLano sin agrupacion
25 xtemp, _ = datasets.make_blobs(n_samples, 2)
26 X[3] = StandardScaler().fit_transform(xtemp)
27
28 #Blobs con deformacion anisotropica
29 xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
30 xtemp = np.dot(xtemp, [[0.6, -0.6], [-0.4, 0.8]])
31 X[4] = StandardScaler().fit_transform(xtemp)
32
33 #blobs con diferentes varianzas
34 xtemp, _ = datasets.make_blobs(n_samples = n_samples, random_state = 43,
35                               cluster_std = [1.0, 2.5, 0.5])
36 X[5] = StandardScaler().fit_transform(xtemp)
37 n_cluster = [2, 2, 3, 3, 3, 3]
38
39 y = []
40 for c, x in zip(n_cluster, X):
41     model = cluster.KMeans(n_clusters=c)
42     model.fit(x)
43     if hasattr(model, 'labels_'):
44         y.append(model.labels_.astype(np.int))
45     else:
46         y.append(model.predict(x))
47 #-----
48 plt.figure(figsize=(27,9))
49 plt.suptitle('KMeans', fontsize = 48)
50 for i in range(6):
51     ax = plt.subplot(2, 3, i+1)
52     ax.scatter(X[i][:,0], X[i][:,1], c=y[i])

```

```

1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from sklearn import cluster, datasets, mixture, metrics
5  from sklearn.preprocessing import StandardScaler
6
7  #creacion de datasets-----
8  np.random.seed(0)
9  n_samples = 1500
10 X = 6 * [None]
11
12 #circulos concentricos
13 xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
14 X[0] = StandardScaler().fit_transform(xtemp)
15
16 #lunas
17 xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
18 X[1] = StandardScaler().fit_transform(xtemp)
19
20 #blobs
21 xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
22 X[2] = StandardScaler().fit_transform(xtemp)
23
24 #Plano sin agrupacion
25 xtemp, _ = datasets.make_blobs(n_samples, 2)
26 X[3] = StandardScaler().fit_transform(xtemp)
27
28 #blobs con deformacion anisotropica
29 xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
30 xtemp = np.dot(xtemp, [[0.6, -0.6], [-0.4, 0.8]])
31 X[4] = StandardScaler().fit_transform(xtemp)
32
33 #blobs con diferentes varianzas
34 xtemp, _ = datasets.make_blobs(n_samples = n_samples, random_state = 43,
35                               cluster_std = [1.0, 2.5, 0.5])
36 X[5] = StandardScaler().fit_transform(xtemp)
37
38 n_cluster = [2,2,3,3,3,3]
39
40 y = []
41 eps = [0.3,0.3,0.3,0.3,0.15,0.18]
42 for e, x in zip(eps, X):
43     model = cluster.DBSCAN(eps=e)
44     #model = cluster.SpectralClustering(n_clusters=c, affinity='nearest_neighbors')
45     #model = cluster.KMeans(n_clusters=c)
46     #model = cluster.Birch(n_clusters=c)
47     model.fit(x)
48     if hasattr(model, 'labels_'):
49         y.append(model.labels_.astype(np.int))
50     else:
51         y.append(model.predict(x))
52 #-----
53 plt.figure(figsize=(27,9))
54 plt.suptitle('DBSCAN', fontsize = 48)
55 #plt.suptitle('KMeans', fontsize = 48)
56 for i in range(6):
57     ax = plt.subplot(2, 3, i+1)
58     ax.scatter(X[i][:,0], X[i][:,1], c=y[i])

```



```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn import cluster, datasets, mixture, metrics
5 from sklearn.preprocessing import StandardScaler
6
7 #creacion de datasets-----
8 np.random.seed(0)
9 n_samples = 1500
10 X = 6 * [None]
11
12 #circulos concentricos
13 xtemp, _ = datasets.make_circles(n_samples=n_samples, factor=0.5, noise=0.05)
14 X[0] = StandardScaler().fit_transform(xtemp)
15
16 #lunas
17 xtemp, _ = datasets.make_moons(n_samples=n_samples, noise=0.05)
18 X[1] = StandardScaler().fit_transform(xtemp)
19
20 #8blobs
21 xtemp, _ = datasets.make_blobs(n_samples, random_state=8)
22 X[2] = StandardScaler().fit_transform(xtemp)
23
24 #Plano sin agrupacion
25 xtemp, _ = datasets.make_blobs(n_samples, 2)
26 X[3] = StandardScaler().fit_transform(xtemp)
27
28 #8blobs con deformacion anisotropica
29 xtemp, _ = datasets.make_blobs(n_samples, random_state=170)
30 xtemp = np.dot(xtemp, [[0.6, -0.6], [-0.4, 0.8]])
31 X[4] = StandardScaler().fit_transform(xtemp)
32
33 #blobs con diferentes varianzas
34 xtemp, _ = datasets.make_blobs(n_samples = n_samples, random_state = 43,
35                               cluster_std = [1.0, 2.5, 0.5])
36 X[5] = StandardScaler().fit_transform(xtemp)
37
38 n_cluster = [2,2,3,3,3,3]
39
40 y = []
41 for c, x in zip(n_cluster, X):
42
43     model = mixture.GaussianMixture(n_components=c, covariance_type='full')
44     #model = cluster.SpectralClustering(n_clusters=c, affinity='nearest_neighbors')
45     #model = cluster.KMeans(n_clusters=c)
46     #model = cluster.Birch(n_clusters=c)
47     model.fit(x)
48     if hasattr(model, 'labels_'):
49         y.append(model.labels_.astype(np.int))
50     else:
51         y.append(model.predict(x))
52
53 #-----
54 plt.figure(figsize=(27,9))
55 plt.suptitle('GaussianMixture', fontsize = 48)
56 #plt.suptitle('KMeans', fontsize = 48)
57 for i in range(6):
58     ax = plt.subplot(2, 3, i+1)
59     ax.scatter(X[i][:,0], X[i][:,1], c=y[i])

```